

## On-line visual correction & makeup systems. 2—Software

P. M. H. Tobias

In editing systems as much text as possible should be displayed while retaining adequate interline spacing and readable characters. This improves editing performance in two ways. If more text is displayed, fewer changes of the reviewed area of text are needed. Each time the viewed area is changed, the operator has to re-adjust himself to the new position of his reading point on the screen, and this slows him down. Also when correcting from a marked proof, which is probably at present the most efficient way of using an editing system, the operator need not read the text, he has only to find the given correction point on the screen. This is most easy when there is a large amount of text either side of the correction point. It is fairly obvious that taking the needs of page makeup into account a minimum of one page should preferably be displayed.

The screened representation of characters in the tape being edited is not such a straightforward subject as it might at first appear. For program organisation it would be preferable to have a one to one correspondence between codes in the edited text and those codes sent to the character generator for display. There are three different sets of characters which present problems—spaces, format instruction codes and ligatures. In the case of ligatures it is probably best to break them down on input and re-form them again at the end of the editing pass. There are various different spaces used in printing—justifying spaces, em, en and thin spaces, and since characters on the screen would not normally be spaced as they are on the proof, it is necessary for the operator to distinguish between them. A blank might be used for the justifying space, and unique characters for the other spaces. Any letterspaced line would then be very difficult to read. In the 4120 program an apostrophe was used for a (justifying) space. Although very small it made the text much more difficult to read. A small character low down in the line would probably have spoilt the appearance of the text less. Format instruction codes will have to be represented by separate characters but where these occur frequently they will detract from the appearance of the text.

In many editing programs, keyboard text pointer control has been used in conjunction with a light pen. In the 4120 programs, keyboard control alone was employed. As in most programs keys were provided to step the pointer by one character or line. The difference in this case was in a fifth key which when operated led the program to expect one of the four marker move keys. When one of these was pressed, the marker would run in the appropriate direction until stopped by the edge of the screen or by striking any key. The running speed could be set by keyboard. With practice in its use this method was found to be quite practical and certainly is capable of being further developed.

When trained operators are being used, keyboard pointer control would become more efficient since the operator would not have to move his hands from the keyboard. A tracker ball or perhaps light pen would, however, probably still be preferable.

With indirect means of indicating position such as keyboard and tracker ball, the visibility of the marker becomes more important than with the light pen. This is because the operator has to follow the movement of the pointer through the text instead of being concerned only with the new pointer position. In the 4120 program an underline was used as text marker and was brightened by being displayed four times more often than the text. A denser character than the underline would probably have been preferable.

### The specification of editing facilities

One of the major factors in the efficiency of any system is the editing functions specified and the way in which they are called into action. The editing processes required are primarily insertions, deletions and moving of words, characters, page heads, page numbers, footnotes and white space.

Insertion of all these character strings is essentially the same process but the way in which they are initiated should vary. Running heads and page numbers must be inserted on every page usually with a systematic change to the material inserted (eg incrementing the page number), and need only be keyed in once. Those character strings involved in correction rather than makeup need not be retained in this way. The operator must be able to insert any codes that could have been keyed in during the preparation of the tape and preferably the keyboard should have the same layout so that there could be some exchange of operators between the preparation and editing stages.

Moving text consists of extracting that text from the screen refresh buffer ready for re-insertion, its subsequent deletion and then re-insertion elsewhere. If the delete command always removes text from the screen buffer into the insertion buffer then moving simply consists of a deletion followed by an insertion. This is in any case probably the best form of the delete command since if any mistake is made in deletion, the text may then be restored by an immediate insertion in the same position, taking only one or two keystrokes.

With these main functions of text insertion and deletion, most editing could be carried out quite easily from the operator's point of view. Since, however, they both involve a movement of



text in the main or screen buffers they take more computer time than simply overwriting the text in the main buffer, where this is practical. Small areas could also be deleted in this way by overwriting with blanks. For transposition and spelling errors it is often easiest for the operator simply to overwrite the incorrect characters so this function should normally be incorporated together with some incentive for the operator to use it.

The feasibility of using keys with multiple meanings was noted in a publication by L. F. Buckland as long ago as 1963. However, the increased number of keys with multiple meanings that would be necessary in a more sophisticated program might well result in a slowing of the operator's speed of editing.

Another very useful facility in makeup would be the ability of the computer to indicate the depth of a given section of text. This will be necessary for deciding where to end a page and perhaps for inserting white space. Normally, the position of the first line of the page being corrected will, even if it is not shown on the screen, be available to the computer so that it could place a pointer to what it expects will form the last line. If satisfied with this position the operator can insert the running head for the next page. A further facility would be necessary to indicate the depth of a given section of text or perhaps, preferably, to indicate the last line of a given depth of text starting from the marker position. This would be needed in making space for insertion of diagrams.

The editing facilities outlined could be taken as sufficient for most requirements. In any particular system, however, there are likely to be editing processes which it is thought might be dealt with most efficiently by forming a new function. Making or saving lines is one example for which a good case might be made.

#### **The movement of text when editing**

In the normal course of work the text will have been corrected only above the present correction point so text below this should ideally be left undisturbed to ensure that the operator's reading speed is not slowed down and the correspondence in layout between screen and proof is not lost. This latter point is particularly important since it will be much more difficult to locate correction points in areas where the line endings have changed. If text has to be moved in bulk as the result of a correction then the text above the correction point should be moved. If a change of line ending is required this would prove to be more troublesome.

Two approaches might be used, and perhaps best in conjunction. One is to justify on a line at a time basis, allowing the operator complete control over re-justification. Using the other method the computer would not try to justify the line in which a correction is being made but would wait until a correction is made lower down in the text. Upon the second correction being called the computer would carry out any necessary re-justification up to this point before carrying out the correction.

Whenever a line ending is changed, the operator will need to check the new ending for hyphenation accuracy. If, as in the second method, a large number of endings may be changed at once, the extent of the area changed will have to be indicated by pointers to the operator. These could remain until he makes a further correction. Whether using line release or block re-justification, the start of any unjustified text will have to be indicated to the operator. Again this is probably best done by a pointer in the margin.

It perhaps needs to be emphasised that the program should try

to regain the former line endings as soon as possible by use of the widest range of justifying space widths permitted. Hyphenation should be avoided as far as possible because of a possibility of an incorrect split and because for any hyphenation introduced two extra characters have to be inserted and this is always a comparatively long process.

### Program writing for display systems

In this, as in any other real time application one of the main concerns of the programmer is to ensure that the computer's response is sufficiently fast. The speed one can obtain depends on the hardware system chosen, and on the organisation of the program.

To be able to change immediately to a new page or to look back over the page previously corrected will require a total of 3 pages in store.

Assuming an average page of 50 lines at 70 characters per line and using 6 bits per character, a machine with an 18-bit word will require about 2,300 words for the text buffers alone. 3,500 words are needed for a machine with a 12 or 13-bit word. With an 18-bit, 8,192 word store, a program for a single display should fit quite comfortably, but a single display is not by itself an efficient way of using the central processor.

The main advantage of using a refresh store for recycling text to the display is that a very large load is then taken off the central processor. The refresh buffer has only one disadvantage apart from its cost—its contents need to be completely renewed when the viewed area of text is changed. This slows down the response to this command, but since the command should be used only once for every page corrected this is a small price to pay for the central processor time gained. An ancillary advantage of the refresh buffer system is the storage space gained if both read and write facilities to the central processor are available. In this case the page of text in the refresh buffer need not be duplicated in main store, saving about 1,200 18-bit words or 1,800 12-bit words, of main core store.

Because of its large size the manner in which the main text buffer is organised has an important effect on the space and time requirements of the program.

There are two main approaches: to have a circular buffer with all text in correct order, or to allow text to be entered out of order while keeping a tag on the position of each section. This second technique is considered because with it there is no longer any need, for example, to move text apart for insertions, which can consume a lot of time. Inserted characters would be placed in the next free area of store and a list of addresses amended to maintain the record of text sequence. This non-sequential approach is less attractive when dealing with large blocks of unstructured text than, for instance, in commercial work or in processing classified advertisements, where a list of at least the lengths of each record must always be kept.

In any circular buffer some space must be available to allow an insertion to take place. When an insertion is called for at a point within the text, space must be made at that point for the inserted characters and to do this all the text between the insertion point and the spare insertion space must be moved aside. In many programs this insertion space has been kept at the head of the buffer but a considerable time saving may be made by keeping it as close as possible to the position the operator has reached in correcting the text. When in a system without refresh buffer an insertion or deletion causes a bulk movement of text between the correction point and the insertion space, very little extra time is required to move the insertion space to this correction point. The insertion space will then be very much nearer a subsequent correction point than if always placed at the head of the buffer. In a system with refresh buffer any insertion will be made into the text in this buffer, causing the text above the correction point to be moved up and a number of characters to overflow from the top of the refresh buffer into the main cyclic buffer.

The characters which overflow will have to be inserted into the main buffer at the end of the post-display page and before the page which is next to be corrected (pre-display page). There is obviously a discontinuity in the text at this point as the page currently being displayed is absent from the main buffer. If the insertion space is kept at this discontinuity then characters overflowing from the screen refresh buffer may be packed into the insertion space without any further movement of text.

As more insertions than deletions will be made, the insertion space will eventually fill up. Before this happens, space should be made at the head of the buffer, by making the reader lag behind the punch for as many characters as are required to restore the insertion space to maximum capacity. If all the text between the insertion space and this space at the head of the buffer is then moved up, the insertion space will be restored to full size. The number of characters to be moved up in this way depends on the size of the post-display area.

This method of using a cyclic text buffer and keeping the insertion space near the point that the operator is correcting will be particularly useful where storage space is scarce—as it is likely to be in a practical system. The non-sequential buffer should use less time when insertions and deletions are made but probably requires more space for the program and certainly will for the text buffer and accompanying sequence record.

### Multiple displays

For appreciable periods of time, the operator while looking at his proof and checking it against the text on the screen will make no demands on the central processor. In systems with a refresh buffer this time will be considerably longer than the time the computer takes to satisfy the operator's requirements, so the computer will be considerably under-employed when servicing only one display.

However, when the operator does make a demand it is often for a long time—of the order of seconds for some demands in a

system with refresh buffer, so it is not possible to use all processor time this way without occasionally making an operator wait for a longer period when a number of operators' commands overlap. If a number of operators were simultaneously to demand the display of a new page there could plainly be a long wait for the last of them. This is an extreme case, but queues could develop to a lesser extent on all commands. The upper limit to the number of displays and operators that could be serviced by one computer, before queueing of operators' demands becomes excessive, is a matter of compromise and probably will only be decided after experience.

To avoid the need for further store modules, it may be necessary to store less than three pages of text. If the restriction on looking back over the previously corrected page is accepted then each page could be marked for output immediately after being corrected and only one page per display in main store would then be required, ie 1,200 18-bit words. Once the page is marked for output, punching will start, but if the operator should wish to look back he will still be able to do so, but only back to the point reached by the punch.

The random distribution of the operator's demands means that a processor servicing displays only will always have some slack time which could be used by some low priority job. The rejustification and format control part of the display program can be used for off-line processing. This part of the program will need to be a sub-routine since it will be called in various editing processes. Using this sub-routine, with the addition of an input and output section of 50 words or so, a program for off-line justification would be complete.

It is quite possible that several off-line justification processes could be run simultaneously with the display work. In deciding the maximum number of operators that may efficiently use the same computer it is not possible to decide how many off-line justification processes may be run, until after experience has been gained with practised operators. It seems sensible to gain experience with the minimum configuration which would demonstrate more or less all operating problems. This configuration will consist of two displays and one off-line justification run.

Many applications of displays have been held up because of software difficulties, mainly those in which graphical displays are needed or in which casual users will be operating the system.

Text editing using a display is very much simpler than manipulating graphical matter since text consists of a string of discrete items in unambiguous order, which is not always true with graphical displays. For the same reason the use of the light pen is much simpler. The talk about man/machine communication using light pen, while being relevant to text editing, is probably far less important when a system is being used by a trained full-time operator than for systems being used only occasionally by a number of people. The instantaneous feedback offered by the CRT display together with the computing power of the central processor have great promise in editing systems and they will certainly be extensively used in the future. Present progress awaits a starting point on which to gain experience of the operator's reaction to editing from a display. It is quite likely that with such a powerful tool available, the editing performance of a full-time operator will surprise those of us who are only used to the casual editing of program tapes.